Seminar 7 – Recap

Prolog.

1. Remove from a list all the elements that occur multiple times. Ex: [1, 2, 3, 2, 1] => [3].

This solution is made of 3 predicates:

* Exists: checks if an element appears in a list
* RemoveElem: removes all occurrences of an element from a list
* Solution: removes the elements that occur multiple times.

%exists(L: list, E:element)

% flow model (I, I)

Exists([H|T], E):- H = E.

Exists([H|T], E):- exists(T, E).

%removeElem(L:list, E:element, R:list)

%flow model (I,I, o)

RemoveElem([], X, []).

RemoveElem([H|T], X, R):- H = X, removeElem(T, X, R).

RemoveElem([H|T], X, [H|R]):- removeElem(T, X, R).

%solution(L:list, R:list)

%flow model(I, o)

Solution([], []).

Solution([H|T], S):- exists(T, H), removeElem(T, H, R), !, solution(R, S).

Solution([H|T], [H|S]):-not(exists(T, H)), solution(T, S).

Solution([1,2,3,2,1], R).

R = [3];1

False

Do you think this is going to work?

RemoveElem([1,2,3,2,1], 2, R).

R = [1, 3, 1];

R = [1, 3, 2, 1];

R = [1, 2, 3, 1];

R = [1,2,3, 2, 1];

RemoveElem([1,2,3,2,1], 2) = [1,3,1]

1 U removeElem([2,3,2,1], 2) = [1,3, 1]

RemoveElem([3, 2, 1], 2) = [3,1]

3 U removeElem([2, 1], 2) =[3, 1]

removeElem([1], 2) = [1]

1 U removeElem([], 2) =[1]

[]

Exists([1,2,3,2,1], 1) => true;

Exists([2, 3, 2, 1], 1) =>

exists([3, 2, 1], 1) =>

exists([2, 1], 1) =>

exists([1], 1) => true;

exists([], 1) => false



%comb(L:list, N:integer, R:list)

%flow model (I, I, o)

Comb([E|\_] ,1, [E]).

Comb([\_|T], N, R):-

Comb(T, N, R).

Comb([H|T], N, [H|R]):-

N > 1,

N1 is N – 1,

Comb(T, N1, R).

Comb([1,3,2], 2, R).

R = [1,3];

R = [1,2];

R = [3, 2];

False

* We only want solutions where the elements are in increasing order.

Comb([E|\_] ,1, [E]).

Comb([\_|T], N, R):-

Comb(T, N, R).

Comb([H|T], N, [H, H1 | R]):-

; H < H1,

N > 1,

N1 is N – 1,

Comb(T, N1, [H1|R]),

H < H1.

Is this version correct?

Lisp

a.

(setq a ‘(1 2 3 4))

(setq b ‘a)

(setq c ‘(length ‘(1 2 3 4)))

(print a) => (1 2 3 4)

(print b) => A

(print c) => (length ‘(1 2 3 4))

C

=> (length ‘(1 2 3 4))

(eval c) => 4

(eval b) => (1 2 3 4)

(eval a) => Error. 1 is not a function name.

b.

(setq car ‘cdr)

(car ‘(1 2 3 4)) => 1

(eval car ‘(1 2 3 4)) => Error. Too many parameters passed to eval.

(eval (cons car ‘(1 2 3 4))) => Error. Too many parameters passed to CDR.

(cdr 1 2 3 4)

(eval (list car ‘(1 2 3 4))) => Error. 1 is not a function name

(cdr (1 2 3 4))

(eval (list car ‘ ’(1 2 3 4))) => (2 3 4)

(cdr ‘(1 2 3 4))

(eval car) => Error. Variable CDR has no value.

> cdr

(apply car ‘(1 2 3 4 5)) <=> (cdr 1 2 3 4 5))

=> Error. Too many params for CDR

(apply #’car ‘(1 2 3 4 5)) <=> (car 1 2 3 4 5)

=> Error.

(funcall car ‘((1 2 3 4 5))) => NIL

(funcall #’car ‘((1 2 3 4 5))) => (1 2 3 4 5)

(funcall car ‘(1 2 3 4 5)) => (2 3 4 5)

(funcall #’car ‘(1 2 3 4 5)) => 1

c.

(mapcar #’list ‘(1 2 3 4 5)) <=> (list (list 1) (list 2) (list 3) (list 4) (list 5)) => ((1) (2) (3) (4) (5))

(mapcan #’list ‘(1 2 3 4 5)) <=> (nconc (list 1) (list 2) (list 3) (list 4) (list 5)) => (1 2 3 4 5)

(maplist #’list ‘(1 2 3 4 5)) <=> (list (list (1 2 3 4 5)) (list (2 3 4 5)) (list (3 4 5)) (list (4 5)) (list (5))) =>

(((1 2 3 4 5)) ((2 3 4 5)) ((3 4 5)) ((4 5)) ((5)))

(mapcon #’list ‘(1 2 3 4 5)) => ((1 2 3 4 5) (2 3 4 5) (3 4 5) (4 5) (5))

(apply #’append (mapcon #’list ‘(1 2 3 4 5 ))) => (1 2 3 4 5 2 3 4 5 3 4 5 4 5 5)

d.

(apply #’max ‘(1 2 3 4 5)) => 5

(mapcar #’max ‘(1 2 3 4 5)) => (1 2 3 4 5)

(eval (append ‘(+) (mapcar #’max ‘(1 2 3 4 5))) => 15

(+ 1 2 3 4 5)

(apply #’+ (mapcar #’max ‘(1 2 3 4 5))) => 15

e.

(mapcar #’(lambda (a b) (eval (list a b)))

‘(list max min evenp)

‘(1 2 3 4 5 6))

=> ((1) 2 3 t)

(lambda list 1) => (list 1) => (1)

(lambda max 2) => (max 2) => 2

(lambda min 3) => (min 3) => 3

(lambda evenp 4) => (evenp 4) => t

f.

(setq x ‘(1 2 3 4 5))

(setq y ‘(6 7 8 9 10 11 12))

(mapcar #’(lambda (a b c d) (eval (funcall c d a b)))

X

Y

(mapcar #’(lambda(q) ‘list) y) ;(list list list list list list list)

(mapcar #’(lambda (v) ‘+) x) ; (+ + + + +)

)

(+ 1 6) => 7

(+ 2 7) ...